Créer un compte Se connecter

Article Discussion Lire Modifier le code Afficher l'historique Rechercher

WIKIPÉDIA
L'encyclopédie libre

Navigation
Accueil
Portails thématiques
Article au hasard
Contact

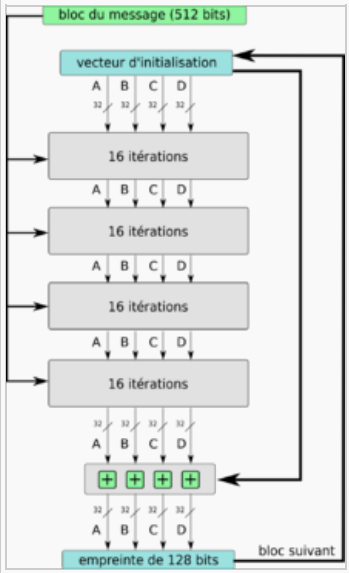
Contribuer
Débuter sur Wikipédia
Aide
Communauté
Modifications récentes
Faire un don

Imprimer / exporter
Créer un livre
Télécharger comme PDF
Version imprimable

Outils
Pages liées
Suivi des pages liées
Importer un fichier
Pages spéciales
Adresse de cette version
Information sur la page
Élément sur Wikidata
Citer cette page

Autres langues
العربية
Български
Català
Čeština
Dansk
Deutsch
English
Esperanto
Español
Euskara
فارسی
Suomi
עברית
Hrvatski
Magyar
Հայերեն
Bahasa Indonesia
Italiano
日本語
한국어
Lietuvių
മലയാളം
Bahasa Melayu
Nederlands
Norsk bokmål
Polski
Português
Română
Русский

MD5



Vue générale de MD5

L'algorithme **MD5**, pour **Message Digest 5**, est une fonction de hachage cryptographique qui permet d'obtenir l'empreinte numérique d'un fichier (on parle souvent de *message*). Il a été inventé par Ronald Rivest en 1991.

L'utilisation de cette fonction de hachage dans les signatures numériques peut conduire à de multiples scénarios d'attaque[1] et n'est plus considérée comme un composant fiable de l'infrastructure à clés publiques. Cependant dans le calcul de la « signature » d'un fichier il reste plutôt fiable, même si l'on ne peut pas assurer qu'il y a unicité entre l'empreinte calculée et le fichier ou message source[2].

Sommaire

1 Historique
2 Exemple
3 Cryptanalyse
4 Algorithme
4.1 Notation
4.2 Préparation du message
4.3 Boucle principale
5 Pseudocode
6 Notes et références
7 Liens externes

Historique [modifier le code]

MD5 (*Message Digest 5*) est une fonction de hachage cryptographique qui calcule, à partir d'un fichier numérique, son *empreinte numérique* (en l'occurrence une séquence de 128 bits ou 32 caractères en notation hexadécimale) avec une probabilité très forte que deux fichiers différents donnent deux empreintes différentes.

En 1991, Ronald Rivest améliore l'architecture de MD4 pour contrer des attaques potentielles qui seront confirmées plus tard par les travaux de Hans Dobbertin.

Cinq ans plus tard, en 1996, une faille qualifiée de « grave » (possibilité de créer des collisions à la demande) est découverte et indique que MD5 devrait être mis de côté au profit de fonctions plus robustes comme SHA-1.

En 2004, une équipe chinoise découvre des collisions complètes. MD5 n'est donc plus considéré comme sûr au sens cryptographique. On suggère maintenant d'utiliser plutôt des algorithmes tels que SHA-256, RIPEMD-160 ou Whirlpool.

Cependant, la fonction MD5 reste encore largement utilisée comme outil de vérification lors des téléchargements et l'utilisateur peut valider l'intégrité de la version téléchargée grâce à l'empreinte. Ceci peut se faire avec un programme comme *md5sum* pour MD5 et *sha1sum* pour SHA-1.

Comme toute fonction de hachage cryptographique, MD5 peut aussi être utilisé pour calculer l'empreinte d'un mot de passe avec la présence d'un *sel* permettant de ralentir une attaque par force brute. Cela a été le système employé dans GNU/Linux. Ainsi, plutôt que de stocker les mots de passe dans un fichier, ce sont leurs empreintes MD5 qui sont enregistrées, de sorte que quelqu'un qui lirait ce fichier ne pourrait pas découvrir les mots de passe. La commande *enable secret* des commutateurs et routeurs Cisco, utilisait le hachage MD5 (5 pour indiquer MD5) pour stocker le mot de passe du mode privilégié dans le fichier de configuration de l'équipement. Les dernières versions d'IOS intègrent le hachage SHA256 (4 pour indiquer SHA256)[3].

Le programme John the ripper permet de casser (trouver une collision pour) les MD5 triviaux par force brute. Il est incommode pour les clés longues, et ne fonctionne pas toujours si elles contiennent des caractères nationaux spécifiques (cela dépend en fait des dictionnaires utilisés).

Les *tables arc-en-ciel* (à accès direct, et qui font parfois plusieurs gigaoctets) permettent de les craquer souvent en moins d'une seconde. Ces tables utilisent des dictionnaires établis après plusieurs jours, mois ou années de calcul. Ceux-ci ne contiennent pas la totalité des clés MD5 possibles, ni ne sont destinés à un cassage par force brute (une empreinte comporte 128 bits, ce qui représente environ 400 sextillions (4.10^{38}) de combinaisons), mais permettent par examen de l'empreinte d'éliminer de très grandes classes de combinaisons à *ne pas* tester, ce qui accélère la recherche plusieurs milliards de fois. L'efficacité des tables arc-en-ciel diminue si l'empreinte est calculée avec un

Simple English
Slovenčina
Slovenščina
Српски / srpski
Svenska
Тоҷикӣ
ไทย
Türkçe
Українська
Tiếng Việt
中文
Modifier les liens

« sel ».

Exemple [modifier le code]

Voici l'empreinte (appelée abusivement *signature*) obtenue sur une phrase :

```
MD5("Wikipedia, l'encyclopedie libre et gratuite") = d6aa97d33d459ea3670056e737c99a3d
```

En modifiant un caractère, cette empreinte change radicalement :

```
MD5("Wikipedia, l'encyclopedie libre et gratuitE") = 5da8aa7126701c9840f99f8e9fa54976
```

Très concrètement, la vérification de l'empreinte ou somme de contrôle MD5 peut être réalisée de la façon suivante : lors du téléchargement d'un programme, on note la série de caractères nommée "Signature MD5" indiquée sur la page de téléchargement. Quand ce téléchargement est terminé, on lance un utilitaire de calcul MD5 comme, par exemple: *HashCalc* ou *md5sums*, qui indique entre autres la somme de contrôle correspondant au fichier. Si les deux valeurs correspondent, on peut alors raisonnablement considérer que le fichier n'a pas été corrompu (volontairement ou non d'ailleurs). On constate plusieurs fragilités dans ce processus : la page d'origine a pu être modifiée, et l'utilitaire de calcul peut être *adapté* pour fournir la signature attendue. C'est pourquoi il faut impérativement utiliser un utilitaire provenant d'une source de confiance. Il est aussi possible d'utiliser une extension pour le navigateur Mozilla Firefox comme *MD Hash tool*[4] afin d'automatiser ce contrôle.

Cryptanalyse [modifier le code]

À ses débuts, la fonction MD5 était considérée comme sûre, mais au cours du temps, des failles potentielles ont été découvertes dans son fonctionnement et durant l'été 2004, il a été cassé par des chercheurs chinois, Xiaoyun Wang, Dengguo Feng, Xuejia Lai (co-inventeur du célèbre algorithme de chiffrement IDEA) et Hongbo Yu. Leur attaque a permis de découvrir une collision complète (deux messages différents qui produisent la même empreinte) sans passer par une méthode de type recherche exhaustive[2,5].

Sur un système parallélisé, les calculs n'ont pris que quelques heures. Le MD5 n'est donc plus considéré comme sûr, mais l'algorithme développé par ces trois chercheurs concerne des collisions quelconques et ne permet pas de réaliser une collision sur une empreinte spécifique, c'est-à-dire réaliser un deuxième message, à partir de l'empreinte d'un premier message, qui produirait la même empreinte. Un projet de calcul distribué lancé en mars 2004, MD5CRK, visait à découvrir une collision complète mais a été subitement arrêté après la découverte de l'équipe chinoise. La sécurité du MD5 n'étant plus garantie selon sa définition cryptographique, les spécialistes recommandent d'utiliser des fonctions de hachage plus récentes comme le SHA-256.

On sait maintenant générer des collisions MD5 en moins d'une minute lorsque les deux blocs en collisions sont « libres »[6]. On peut aussi générer une infinité de collisions avec un texte T à partir de deux messages M1 et M2 de même longueur qui sont en collision. Il suffit de concaténer M1 et M2 avec T, tel que T1 = M1 + T et T2 = M2 + T, afin d'obtenir une collision complète entre T1 et T2. On ne peut toutefois pas générer une signature particulière et la falsification de documents reste un exercice difficile.

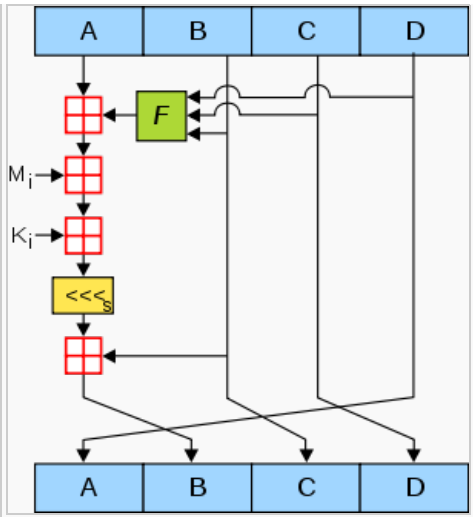
Dès 2006, il est par exemple possible de créer des pages HTML aux contenus très différents et ayant pourtant le même MD5. La présence de métacodes de « bourrage » placés en commentaires, visibles seulement dans la source de la page web, trahit toutefois les pages modifiées pour usurper le MD5 d'une autre. La supercherie peut donc être levée si on examine les sources de la page en question.

En 2008, le logiciel BarWF[7] utilise les ressources des instructions SSE2 et des processeurs massivement parallèles d'une carte graphique (CUDA) pour casser du MD5 en force brute à la vitesse annoncée de 350 millions de clés par seconde.

Algorithme [modifier le code]

Notation [modifier le code]

- $[<<<]_s$ est une rotation de *s* bits vers la gauche, *s* varie pour chaque opération.
- [+] symbolise l'addition modulo 2^{32}.
- $\oplus, \wedge, \vee, \neg$ symbolisent respectivement les opérations booléennes XOR, AND, OR et NOT.



Une opération de MD5. MD5 comprend 64 blocs de ce type, groupés en quatre tours de 16 opérations. *F* est une fonction non-linéaire, qui varie selon le tour. M_i symbolise un bloc de 32 bits provenant du message à hacher et K_i est une constante de 32 bits, différentes pour chaque opération.

Préparation du message [modifier le code]

MD5 travaille avec un message de taille variable et produit une empreinte de 128 bits. Le message est divisé en blocs de 512 bits, on applique un remplissage de manière à avoir un message dont la longueur est un multiple de 512. Le remplissage se présente comme suit :

- on ajoute un '1' à la fin du message
- on ajoute une séquence de '0' (le nombre de zéros dépend de la longueur du remplissage nécessaire)
- on écrit la taille du message, un entier codé sur 64 bits

Ce remplissage est toujours appliqué, même si la longueur du message peut être divisée par 512. Cette méthode de *padding* est semblable à celle utilisée dans la plupart des algorithmes de *Message Digest* des familles MD (comme MD5 ou RIPEMD) ou SHA (SHA-1 ou SHA-512) mais différente de celle de l'algorithme Tiger qui utilise une convention dite Little endian d'ordonnancement des bits dans chaque octet.

La taille du message est codée en Little endian. Le message a maintenant une taille en bits multiple de 512, c'est-à-dire qu'il contient un multiple de 16 mots de 32 bits.

Boucle principale [modifier le code]

L'algorithme principal travaille avec un état sur 128 bits. Il est lui-même divisé en 4 mots de 32 bits : *A*, *B*, *C* et *D*. Ils sont initialisés au début avec des constantes. L'algorithme utilise ensuite les blocs provenant du message à hacher, ces blocs vont modifier l'état interne. Les opérations sur un bloc se décomposent en quatre rondes (étapes), elles-mêmes subdivisées en 16 opérations similaires basées sur une fonction non-linéaire *F* qui varie selon la ronde, une addition et une rotation vers la gauche. Les quatre fonctions non-linéaires disponibles sont :

$$F(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$$
$$G(B,C,D) = (B \wedge D) \vee (C \wedge \neg D)$$
$$H(B,C,D) = B \oplus C \oplus D$$
$$I(B,C,D) = C \oplus (B \vee \neg D)$$

Pseudocode [modifier le code]

MD5 peut s'écrire sous cette forme en pseudo-code.

```
//Note: Toutes les variables sont sur 32 bits

//Définir r comme suit :
var entier[64] r, k
r[ 0..15] := {7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22}
r[16..31] := {5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20}
r[32..47] := {4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23}
r[48..63] := {6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21}

//MD5 utilise des sinus d'entiers pour ses constantes:
pour i de 0 à 63 faire
    k[i] := floor(abs(sin(i + 1)) × 2^32)
fin pour

//Préparation des variables:
var entier h0 := 0x67452301
var entier h1 := 0xEFCDAB89
var entier h2 := 0x98BADCFE
var entier h3 := 0x10325476

//Préparation du message (padding) :
ajouter le bit "1" au message
```

```
ajouter le bit "0" jusqu'à ce que la taille du message en bits soit égale à 448 (mod 512)
ajouter la taille du message codée en 64-bit little-endian au message

//Découpage en blocs de 512 bits:
pour chaque bloc de 512 bits du message
    subdiviser en 16 mots de 32 bits en little-endian w[i], 0 ≤ i ≤ 15

    //initialiser les valeurs de hachage:
    var entier a := h0
    var entier b := h1
    var entier c := h2
    var entier d := h3

    //Boucle principale:
    pour i de 0 à 63 faire
        si 0 ≤ i ≤ 15 alors
              f := (b et c) ou ((non b) et d)
              g := i
        sinon si 16 ≤ i ≤ 31 alors
              f := (d et b) ou ((non d) et c)
              g := (5×i + 1) mod 16
        sinon si 32 ≤ i ≤ 47 alors
              f := b xor c xor d
              g := (3×i + 5) mod 16
        sinon si 48 ≤ i ≤ 63 alors
            f := c xor (b ou (non d))
            g := (7×i) mod 16
        fin si
        var entier temp := d
        d := c
        c := b
        b := ((a + f + k[i] + w[g]) leftrotate r[i]) + b
        a := temp
    fin pour

    //ajouter le résultat au bloc précédent:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
fin pour

var entier empreinte := h0 concaténer h1 concaténer h2 concaténer h3 //(en little-endian)
```

Notes et références [modifier le code]

1. ↑ (en) Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger, « Creating a rogue CA certificate », 30 décembre 2008
2. ↑ a et b (en) « Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD » [PDF]
3. ↑ Wendell Odom, *Préparation à la certification CCNA ICND1 et CCENT : Guide de la préparation officiel Deuxième édition*, PEARSON, 2007, CCIE 1624 éd., 662 p. (ISBN 978-2-7440-7285-7), chap. 9 (« Configuration des commutateurs Ethernet »), p. 265
   à voir en première ligne de la page
4. ↑ « mozdev.org - mdhashtool: installation »
5. ↑ (en) « Musings on the Wang etal. MD5 Collision » [PDF]
6. ↑ Vlastimil Klima, « Tunnels in Hash Functions: MD5 Collisions Within a Minute, Cryptology ePrint Archive Report 2006/105 », 17 avril 2006
7. ↑ (en) « World Fastest MD5 cracker BarsWF »

Liens externes [modifier le code]

- (en) « RFC 1321 qui détaille l'algorithme »
- « Fonction de décryptage MD5 »
- (en) Ondrej Mikle, « Practical Attacks on Digital Signatures Using MD5 Message Digest »
- (en) Vlastimil Klíma, « Finding MD5 collisions - A toy for notebook » [PDF]
- (en) P. Hawkes *et al.*, « Musings on the Wang et al. - MD5 Collision » [PDF]

v · d · m **Fonctions de hachage cryptographiques**

Algorithmes	APR1 · AR · Boognish · FFT-hash · HAS-160 · Haval · MD2 · MD4 · **MD5** · MD6 · N-hash · PANAMA · RIPEMD · RIPEMD-128 · RIPEMD-160 · RIPEMD-256 · SHA-0 · SHA-1 · SHA-2 (SHA-224 · SHA-256 · SHA-384 · SHA-512) · SHA-3 · Snefru · StepRightUp · Tiger · VSH · Whirlpool
Cryptanalyse	Paradoxe des anniversaires · Linéaire · Différentielle · Attaque par force brute · Effet avalanche · Pseudo-collision
Architecture	Remplissage · Fonction de compression · Construction de Merkle-Damgård · Construction de Miyaguchi-Preneel · Construction de Matyas-Meyer-Oseas · Construction de Davies-Meyer · Construction de l'éponge

Portail de la cryptologie **Portail de la sécurité informatique**

Catégorie : Algorithme de hachage

Dernière modification de cette page le 6 mars 2014 à 17:53.

Droit d'auteur : les textes sont disponibles sous licence Creative Commons paternité partage à l'identique ; d'autres conditions peuvent s'appliquer. Voyez les conditions d'utilisation pour plus de détails, ainsi que les crédits graphiques. En cas de réutilisation des textes de cette page, voyez comment citer les auteurs et mentionner la licence.
Wikipedia® est une marque déposée de la Wikimedia Foundation, Inc., organisation de bienfaisance régie par le paragraphe 501(c)(3) du code fiscal des États-Unis.

Politique de confidentialité À propos de Wikipédia Avertissements Développeurs Version mobile